

High Performance Time Series Quantitative Retrieval from Satellite Images on a GPU Cluster

Jia Liu, Yong Xue, *Senior Member IEEE*, Kaijun Ren, Junqiang Song, Christopher Windmill, Patrick Merritt

Abstract—The quality and accuracy of remote sensing instruments continue to increase, allowing geoscientists to perform various quantitative retrieval applications to observe the geophysical variables of land, atmosphere, ocean etc. The explosive growth of time-series RS data over large-scales poses great challenges on managing, processing and interpreting RS “Big Data”. To explore these time series RS data efficiently, in this paper, we design and implement a high performance framework to address the time-consuming time-series quantitative retrieval issue on a GPU cluster, taking the aerosol optical depth (AOD) retrieval from satellite images as a study case. The presented framework exploits the multi-level parallelism for time-series quantitative RS retrieval to promote efficiency. At the coarse-grained level of parallelism, the AOD time series retrieval is represented as multi-directed acyclic graph (DAG) workflows and scheduled based on a list-based heuristic algorithm, heterogeneous earliest finish time (HEFT), taking the idle slot and priorities of retrieval jobs into account. At the fine-grained level, the parallel strategies for the major remote sensing image processing algorithms divided into three categories, i.e. the point or pixel-based operations, the local operations and the global or irregular operations have been summarized. The parallel framework was implemented with message passing interface (MPI) and compute unified device architecture (CUDA), and experimental results with the AOD retrieval case verify the effectiveness of presented framework.

Index Terms—Quantitative remote sensing retrieval; time series; aerosol optical depth; GPU Cluster; high performance computing; multi-level parallelism; multi-DAG scheduling.

I. INTRODUCTION

In recent decades, the volume of Earth observation data has increased rapidly. The datacenters of the China national satellite meteorological center (NSMC) have archived 4.126 PBs of data, and the China center for resources satellite data and application (CCRS DA) achieved more than 16 million scenes of remote sensing (RS) images up to August 2017 as reported [1]. As a typical example, data from the moderate resolution imaging spectroradiometer (MODIS) instruments onboard the satellite TERRA and AQUA have been used to study the properties of land, atmosphere, ocean widely since been launched in December 1999 and May 2002 respectively. Each MODIS sensor produces 70 GB of raw data per day [2], from which large amount of higher-level products have been generated by national aeronautics and space administration’s MODIS adaptive processing system (NASA’s MODAPS). Great efforts have been made to develop quantitative remote sensing models to estimate various geophysical parameters such as normalized difference vegetation index (NDVI) and aerosol optical depth (AOD), and link these massive data to different aspects of dynamic earth [3, 4]. Time series RS data covering large areas have been extensively used to monitor temporal and spatial changes and patterns of Earth [5, 6]. For instance, MODIS AOD dataset are used to portray the global, regional, and seasonal distribution of AOD [7, 8]. However, the large amount of archived RS data and the relatively poor performance of compute-intensive algorithms can reduce the data utilization and delay the response [9, 10], which raise the need for effective processing methods of RS data [11].

High performance computing (HPC) technologies have been incorporated into the RS community in recent years to address the efficiency needs, and the efforts generally fall into three categories [12], i.e. the data processing based on the specialized hardware such as graphics processing units (GPUs) [13] and field programmable gate arrays (FPGAs), the data processing on a cluster [12, 14] and large-scale distributed computing infrastructures such as Grid [15] and Cloud [16]. During the past several years, GPU has evolved into a highly parallel, multithreaded, many-core processor

This work was supported in part by the National Natural Science Foundation of China (grant no. 61572510, 41471306 and 41590855), the Ministry of Science and Technology of China (grant no. 2016YFC0200500) and the Strategic Priority Research Program of the Chinese Academy of Sciences (grant no. XDA19080303). (*Corresponding authors: Professor Yong Xue.*)

J. Liu is with the School of Computer Science, China University of Geosciences (Wuhan), Wuhan 430074, China and also with the College of Meteorology and Oceanography, National University of Defense Technology, Changsha 410073, China (e-mail: liujia2016@nudt.edu.cn).

Y. Xue is with the School of Environment Science and Spatial Informatics, University of Mining and Technology, Xuzhou, Jiangsu 221116, PR China, and also with the Department of Electronics, Computing and Mathematics, College of Engineering and Technology, University of Derby, Kedleston Road, Derby DE22 1GB, UK (e-mail: y.xue@derby.ac.uk).

K.J. Ren and J.Q. Song are with the College of Meteorology and Oceanography, National University of Defense Technology, Changsha 410073, China (e-mail: renkaijun@nudt.edu.cn; junqiang@nudt.edu.cn).

C. Windmill and P. Merritt are with the Department of Electronics, Computing and Mathematics, College of Engineering and Technology, University of Derby, Kedleston Road, Derby DE22 1GB, UK (e-mail: c.windmill@derby.ac.uk; p.merritt@derby.ac.uk).

with tremendous computational horsepower and very high memory bandwidth, which has been used to accelerate some remote sensing applications. Many research demonstrated that hyperspectral RS data processing such as hyperspectral unmixing, band selection, image classification, automatic target detection [17-20] and real-time missions such as oil spill detection [21], on-board image data processing [22, 23] can benefit from GPUs. In addition, a few efforts have been made to accelerate quantitative RS retrieval applications, in which in general complex partial differential equations need to be solved. Abouali et al. put forward an efficient surface energy balance system (SEBS) algorithm available on GPUs [24]. Su et al. developed a GPU accelerated approach for the electromagnetic scattering using a double-layer vegetation model [25]. Mielikainen et al. implemented a GPU-based parallel radiative transfer model for the infrared atmospheric sounding interferometer [26].

Today GPUs have been integrated into supercomputers as significant components, e.g. the supercomputer Summit at Oak Ridge National Laboratory (ORNL), which captures the NO.1 spot among TOP500 list, has 4,356 nodes. Each node is equipped with two 22-core Power9 CPUs and six NVIDIA Tesla V100 GPUs [27]. Nevertheless, extra efforts need to be paid considering RS applications and the target platform to effectively boost the performance. For instance, Agathos et al. presented a parallel minimum volume simplex analysis algorithm for the hyperspectral unmixing on a multi-GPU platform [28]. Hossam et al. proposed parallel solutions for the recursive hierarchical segmentation analysis on a hybrid CPU/GPU cluster [29]. Lei et al. put forward the image orthorectification processing on a CPU/GPU cluster, which allows flexibility in workload balancing between appropriate computation units [30]. To migrate the time-consuming RS applications to systems with multiple GPUs, research issues including the storage and management of massive RS data, the loading and transmission of RS big data, the scheduling of data-dependent tasks and the efficient programming of RS applications on parallel systems etc. are all challenging [31]. Among these, to explore and exploit multi-level parallelism of applications on a GPU cluster is of great importance to solve the efficiency problem, however, very few work has addressed this issue in the RS community [32].

In this paper, we present a high performance framework to address the time-consuming time series quantitative retrieval applications on a GPU cluster environment, taking the AOD retrieval from satellite image data as a study case. This work investigates the efficient solutions to supports a project to derive a 10-year AOD dataset at 1 km spatial resolution over Asia based on the “synergetic retrieval of aerosol properties model from MODIS data” (SRAP-MODIS) algorithm [33]. The main contributions of this study are as follows: a) design and implement a high performance framework that exploits the multi-level parallelism for the time-series quantitative RS retrieval applications to enhance efficiency; b) develop a coarse-grained parallelism for the AOD time series retrieval which can be represented as multi-directed acyclic graph (DAG) workflows based on a list-based heuristic algorithm,

i.e. the heterogeneous earliest finish time (HEFT), taking the idle slot and priorities of retrieval jobs into consideration; c) the fine-grained parallelism available on GPUs are realized based on our earlier work in [10, 34], and we summarized the parallel strategies for the major remote sensing image data processing algorithms divided into three categories, i.e. the point or pixel-based operations, the local operations, and the global or irregular operations [35]; d) the parallel framework was implemented based on message passing interface (MPI) and compute unified device architecture (CUDA), and the experimental results with the AOD retrieval case verify the effectiveness of the presented framework.

The rest of this paper is organized as follows. Section 2 describes the study case of time series AOD retrieval from satellite image data. Section 3 presents the high performance framework for the time series quantitative RS retrieval on a GPU cluster. Section 4 gives and discusses the experimental results. Conclusions are drawn in Section 5.

II. AEROSOL RETRIEVAL: A CASE STUDY

Aerosols are liquid or solid particles suspended in the air from natural or anthropogenic origins, and are intricately linked to the climate system and to the hydrologic cycle [36]. They have a negative impact on the quantitative retrieval, and influence climate and weather in direct, indirect and semi-direct effect ways [37, 38]. AOD, which is defined as the vertical integration of the aerosol extinction coefficient from the ground to the top of the atmosphere, is a crucial parameter reflecting the amount of aerosols and can be used in further studies such as atmospheric corrections of RS data, and particulate matter retrieval for air pollution. Compared with ground measurements, satellite images can provide an effective tool to detect the temporal-spatial distribution and variation trends because of their large spatial coverage and reliable repeated measurements.

The AOD retrieval procedure based on the SRAP-MODIS algorithm, the case study in this paper, is depicted in Fig. 1, which includes processing modules of data extraction, cloud mask, absorption correction, image cut, geometric correction, region of interest (ROI) acquisition, data interpolation and retrieval [10, 33].

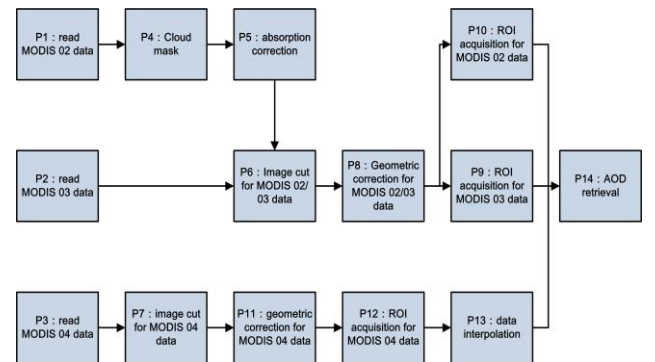


Fig. 1. The AOD retrieval procedure using the SRAP-MODIS algorithm.

- *Data extraction*: Two mid-infrared channels and three visible channels are extracted from MODIS hierarchical

data format (HDF) data with the open source library Geospatial Data Abstraction Library (GDAL), HDF4 and PROJ.4 Cartographic Projections.

- *Cloud mask*: It is performed over land using the spatial variability of $0.47\ \mu\text{m}$ (>0.0025) and $1.38\ \mu\text{m}$ (>0.003) channel reflectance, and the absolute value of $0.47\ \mu\text{m}$ reflectance (>0.4) and $1.38\ \mu\text{m}$ reflectance (>0.025), and then generated through the union of four cloud mask tests above. The spatial variability is calculated as the absolute standard deviation of the reflectance of a 3×3 pixel window.
- *Absorption correction*: It eliminates absorption effects from H_2O , CO_2 , and O_3 on the apparent reflectance with national centers for environmental prediction (NCEP) data. The gas transmission factors are calculated as a function of wavelength, the air mass factor and some weighting coefficients for each pixel in the MODIS L1B data. The reflectance is then corrected by multiplying by the gas transmission factors.
- *Image cut*: Image scenes are resized according to the requested geographic coordinates.
- *Geometric correction*: Since the latitude and longitude data has been provided with the MODIS reflectance data, these can be used to implement the geometric correction using the triangulation warping method with the nearest neighbor resampling. The map projection adopted is the Geographic Lat/Lon projection.
- *ROI acquisition*: Image scenes are mosaicked to obtain large ROIs.
- *Data interpolation*: The input data of the AOD retrieval step are interpolated with the inverse distance weighting and resized with the bilinear interpolation to keep the resolution consistent.
- *AOD retrieval*: It is performed with the SRAP-MODIS algorithm that employs a set of non-linear equations on channel $0.47\ \mu\text{m}$, $0.55\ \mu\text{m}$, and $0.66\ \mu\text{m}$ from Terra and Aqua MODIS to derive the wavelength exponent and the Angstrom's turbidity coefficient and then calculates AOD for each pixel. The accuracy has been validated in earlier work [33, 39] and used to generate AOD datasets over China.

III. METHODOLOGY

A. Framework overview

Fig. 2 presents an overview of the high performance time series quantitative retrieval framework in this paper, which can be represented into three layers, i.e. the data access layer consisted of the MongoDB database and parallel file system, the fine-grained parallelism layer including the coordinator and the parallel executor, and the coarse-grained parallelism layer mainly including the runtime estimator and scheduler from the bottom up. The interaction of the major components is described as follows:

- At the coarse-grained parallelism layer, when the data operations and time-series quantitative retrieval requests are submitted through portable batch system (PBS) with

parameter scripts, the data operation module perform the query, insert, update and delete and metadata update of RS data, and the runtime estimator module predicts tasks' runtime using historical logs to feed the scheduler. The scheduler deployed generates and maintains a list of scheduling results for multi-DAG workflows. This layer realizes a coarse-grained parallelism between jobs, i.e. the retrieval requests with different inputs or parameter settings, and tasks, i.e. the different processing steps in a retrieval workflow job.

- At the fine-grained parallelism layer, the coordinator on the master node receives the scheduling results from the scheduler and generates task assignment scripts for each slave with corresponding parameter settings. This layer achieves a fine-grained data parallelism for processing steps using GPU accelerators.
- At the data access layer, the satellite orbit stripe or scene organization model is adopted. The satellite orbit data is organized according to its reception time, and stored in a scene unit. Each scene is identified by the upper, lower, left and right latitude and longitude coordinates. RS data and metadata are managed with the BeeGFS parallel file system and the MongoDB NoSQL database respectively. The metadata in MongoDB database includes categories of identification, reference system, application, data quality, data distribution and restriction.

B. Multi-level parallelism

The time-series retrieval applications contain multi-level parallelism:

- *Retrieval job level parallelism*: to retrieval a time series AOD product, the request can be considered as multiple retrieval jobs with different data inputs and parameter settings. Each job can be represented as an independent DAG workflow, allowing a coarse-grained parallelism.
- *Task level parallelism*: a retrieval job is consisted of RS data processing steps, such as the data extraction, cloud mask etc. These steps are considered as tasks of a job with control or data dependency and can be represented as a DAG, which indicates a coarse-grained functional parallelism.
- *RS Image processing level parallelism*: the calculation of a pixel, a local region or an irregular part of the RS image data can be decomposed from the spatial domain generally and assigned to multiple threads on GPUs for concurrent execution.

Meanwhile we consider hierarchical parallelism of a GPU cluster as follows in this work:

- *Inter-node parallelism*: it depends on a coarse-grained parallelism, exploiting the concurrency among nodes to parallelize the execution of tasks, i.e. processing steps from retrieval jobs. This level of parallelism is carried out with MPI.
- *Intra-node parallelism*: it is based on a fine-grained parallelism in order to exploit concurrency on GPUs. This strategy uses the concurrent threads with CUDA to accelerate steps.

To obtain high performance, the time-series retrieval is decomposed and mapped onto a GPU cluster, in which each node is equipped with a GPU in this work. The job and task level parallelism are achieved with multi-DAG scheduling

and corresponds to the inter-node parallelism, while the RS image data processing level parallelism is implemented with intra-node strategy.

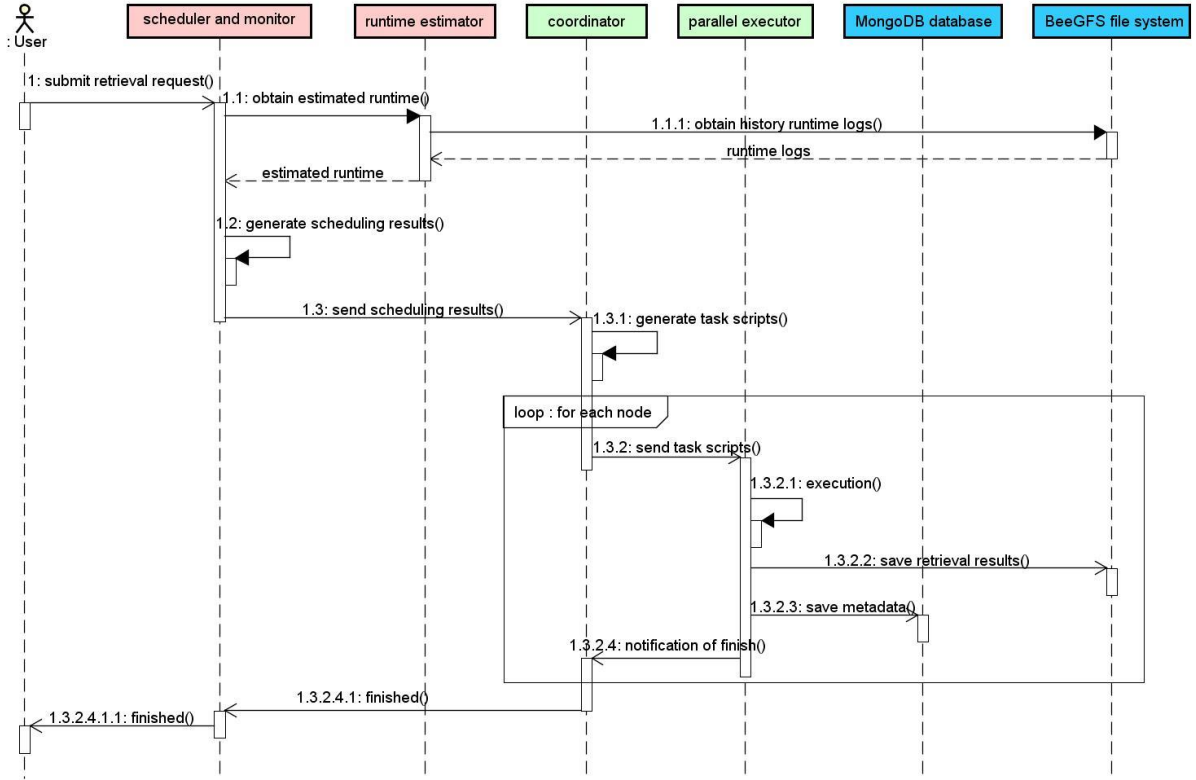


Fig. 2. The overview of the presented high performance framework for the time series quantitative RS retrieval on a GPU cluster.

C. Coarse-grained parallelism

1) Runtime estimation

Empirical investigations indicate that runtime estimated from users is inaccurate [40, 41]. Therefore in this work, we estimate tasks' runtime from historical information. Given that the computing platform configuration is pre-determined, the runtime is characterized, for a specific processing step, as a function of dataset parameter, i.e. the image scenes and ROI size. These input features have a great influence on the performance of the estimated runtime, and meanwhile can be obtained before task running. Linear regression models were constructed for each step as follows:

$$y = \theta^T \mathbf{x} \quad (1)$$

Where $\mathbf{x}^T = [1, x_0]$, and x_0 represents the image scenes for P1~P12, the ROI size for P13 and P14, while y represents the runtime. $\theta = [\theta_0, \theta_1]^T$ are the model parameters and derived with the least squares method by minimizing the cost $J_{LS}(\theta)$ in Equation 2 with historical logs. The runtime unknown for steps is then estimated according to Equation 1.

$$\hat{\theta}_{LS} = \arg \min_{\theta} J_{LS}(\theta) = \arg \min_{\theta} \frac{1}{2} \sum_{i=1}^n (\theta^T \mathbf{x}_i - y_i)^2 \quad (2)$$

2) Task scheduling strategy

A retrieval workflow in Fig. 1 is represented as a DAG in Fig. 3 and can be scheduled based on a list-based heuristics algorithm HEFT, which provides good performance with a low scheduling time [42, 43].

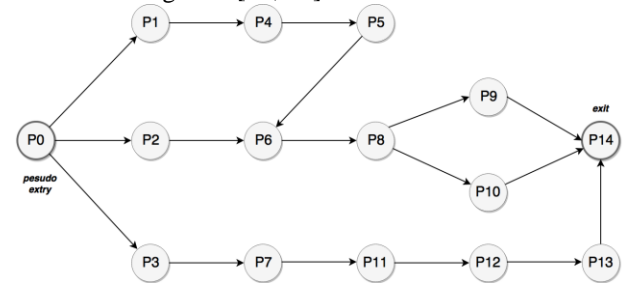


Fig. 3. The DAG diagram for the AOD retrieval procedure in Fig. 1.

The HEFT scheduling algorithm is for a bounded number of heterogeneous processors. It has two stages:

- *The task prioritizing stage:* set the priority of each task with the upward rank $rank_u$ recursively via traversing the task graph upward as shown in Equation 3, where $succ(n_i)$ is the set of immediate successors of task n_i , while \bar{w}_i is the average execution time of a task and $\overline{c_{i,j}}$ is the communication cost for an $edge(i, j)$ which stands for the precedence constraint that task n_i need complete its execution before task n_j starts. The $rank_u$ is defined

as the length of the critical path, i.e. the longest path from the task n_i to the exit, including the computational cost of the node. In a DAG, a task without any parent is called an entry, and a task without any child is called an exit. The rank of exit equals the average computation cost. The task list is then generated by sorting tasks in a decreasing order of $rank_u$. Ties are broken randomly.

- *The processor selection stage*: it follows an insertion based policy, i.e. a task n_i can be inserted to an earliest idle time slot between two already-scheduled tasks on a processor p_j when meeting Equation 4, where the $list_{j,k}$ is the k_{th} task that was already assigned on p_j . Earliest execution start time (EST) and earliest execution finish time (EFT) are two significant attributes.

$$rank_u(n_i) = \bar{w}_i + \max_{n_j \in succ(n_i)} (\bar{c}_{i,j} + rank_u(n_j)) \quad (3)$$

$$EST(list_{j,k+1}, p_j) - EFT(list_{j,k}, p_j) \geq w_{i,j} \quad (4)$$

When tasks in a DAG have been scheduled, the schedule length is the actual finish time (AFT) of the exit task n_{exit} . If there are multiple exits and no pseudo exit task is applied, the makespan is defined as Equation 5. The objective function of HEFT is to determine the assignment of tasks in a DAG to processors such that its schedule length is minimized.

$$makespan = \max\{AFT(n_{exit})\} \quad (5)$$

An example of scheduling result of a daily AOD retrieval workflow with the HEFT algorithm is shown in Fig. 4.

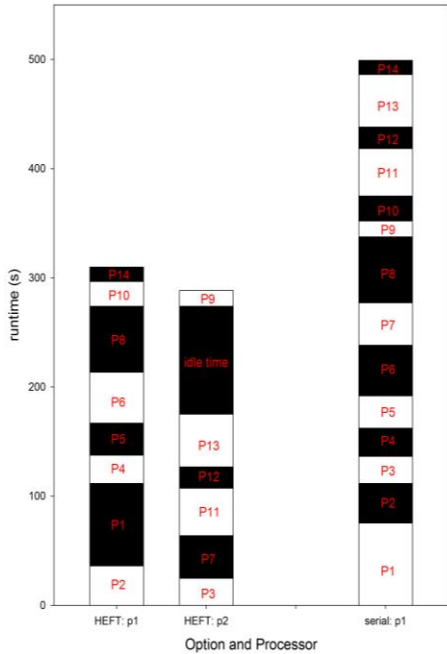


Fig. 4. The scheduling result of a retrieval workflow with HEFT algorithm.

In this work, the AOD time series retrieval is represented as multi-DAG workflows, and we extended the HEFT with the following strategies to realize the coarse-grained level of parallelism:

- *Inter-DAG strategy*: since the HEFT was designed for a single DAG, when scheduling more than one DAG, strategies generally adopted to extend HEFT includes: a) schedule DAGs independently one after another; b)

schedule DAGs interleaving parts of each DAG being scheduled; c) merge the DAGs into a single one and schedule the resulting DAG. Investigations have shown that the interleave and the group strategies can result in better fairness among DAGs, however relatively larger average makespan for the first workflows [44], i.e. the retrieval jobs with higher priorities in this work. Hence, the simple but practical sequential manner was adopted to extend the HEFT algorithm for scheduling multiple DAGs independently.

- *Idle slot search strategy*: it should be noted that there are idle time slots between already schedule tasks on a processor because of the inter-task dependency and the data communication costs between different resources as shown in Fig. 4. Therefore in this work, we adopt an idle slot search strategy to efficiently utilized resources to improve the performance of multi-DAG scheduling.
- *Priority strategy*: consider a common situation that in a batch of jobs for the time-series retrieval parts of jobs might be more urgent for geoscientists. For example, to retrieve AOD results for specific days for monitoring and analyzing heavy air pollution or forest fires. Thus a priority strategy is adopted allowing users to set static priorities along with the retrieval parameter scripts. In this paper, job queues at three levels were constructed according to the static priorities setup by users.

D. Fine-grained parallelism

Most RS image processing algorithms could be classified into three major categories [35]: i.e. the point or pixel-based operations, the local operations and the global or irregular operations as shown in Fig. 5. The fine-grained parallelism is realized on the basis of our previous work in [10, 34].

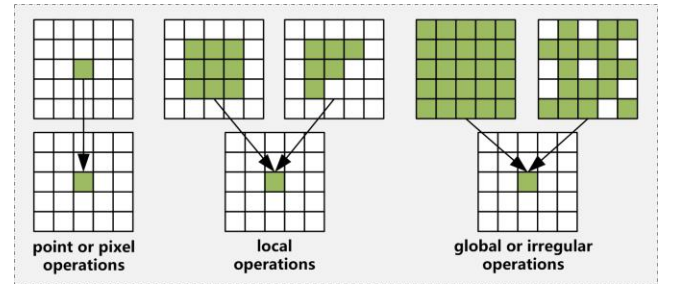


Fig. 5. RS image data processing parallel analysis from the spatial domain.

For the point or pixel-based operations, the computation is performed on each pixel without requiring the context, such as the steps absorption correction and AOD retrieval in this paper. This category most beneficial to the parallelism, since the computation of each pixel can be assigned to a thread of GPUs and needs no communication between. The second category includes the cloud mask and geometric correction in this work. The former involves the convolution operations with a 3*3 pixel window, and the computation of each pixel can be assigned to a thread requiring neighborhood data. The boundary pixels of images need to be dealt with separately after the kernel is finished. The geometric correction shares transformation coefficients and performs resampling for a

triangle, thus the computation of a triangle rather than a pixel is assigned to a thread. The data interpolation is in the last category. The calculation of a pixel in the output image is linked to the overall or irregular parts of the image and can be assigned to a thread.

To tune the fine-grained parallel performance on a single GPU, major optimizations have been applied. Occupancy is the ratio of active warps on a stream multiprocessor (SM) to the maximum number of active warps supported by a SM, and it normally should be improved to make sure that more threads are executing on each SM [45]. However, existing investigations have shown that higher occupancy does not always lead to better performance [46, 47]. Therefore the impacts of registers used and thread-block configuration on the runtime have been analyzed and tuned for each kernel, as stated in our previous work in [10, 34]. Besides, we improve the memory allocation and I/O transfer. The created thread distribution among image data leads to hardware adjusted and coalesced memory accesses for the global memory, and in kernels such as the ‘CloudMask’ and ‘InverseDistance’, the dynamic and static shared memory with higher memory bandwidth are used for sharing data per thread block.

E. Hybrid MPI-CUDA implementations

Major steps such as the cloud mask, absorption correction, geometric correction, data interpolation and AOD retrieval have been accelerated on GPUs with the runtime profiling of AOD retrieval in our previous work [10]. Fig. 6 shows the procedures on the CPU host and GPU device, and the data transfer, the kernels, and the data input/output (I/O). The red arrows represent reading data, the green ones are for writing, and the blue ones stands for the data I/O together with the data transfer between CPU and GPU.

The coarser-grained parallelism is implemented based on MPI with a master-slave mode. The pseudo code is shown in Algorithm 1. The master obtains the job list with parameter settings, performs the scheduling algorithm in Algorithm 2 to generate the scheduling results in the *schedule[job][task]*, generates and distributes local scheduling lists for each slave node in *job_task_list[index_proc]*. The slaves perform the fine-grained parallelism with the local scheduling lists.

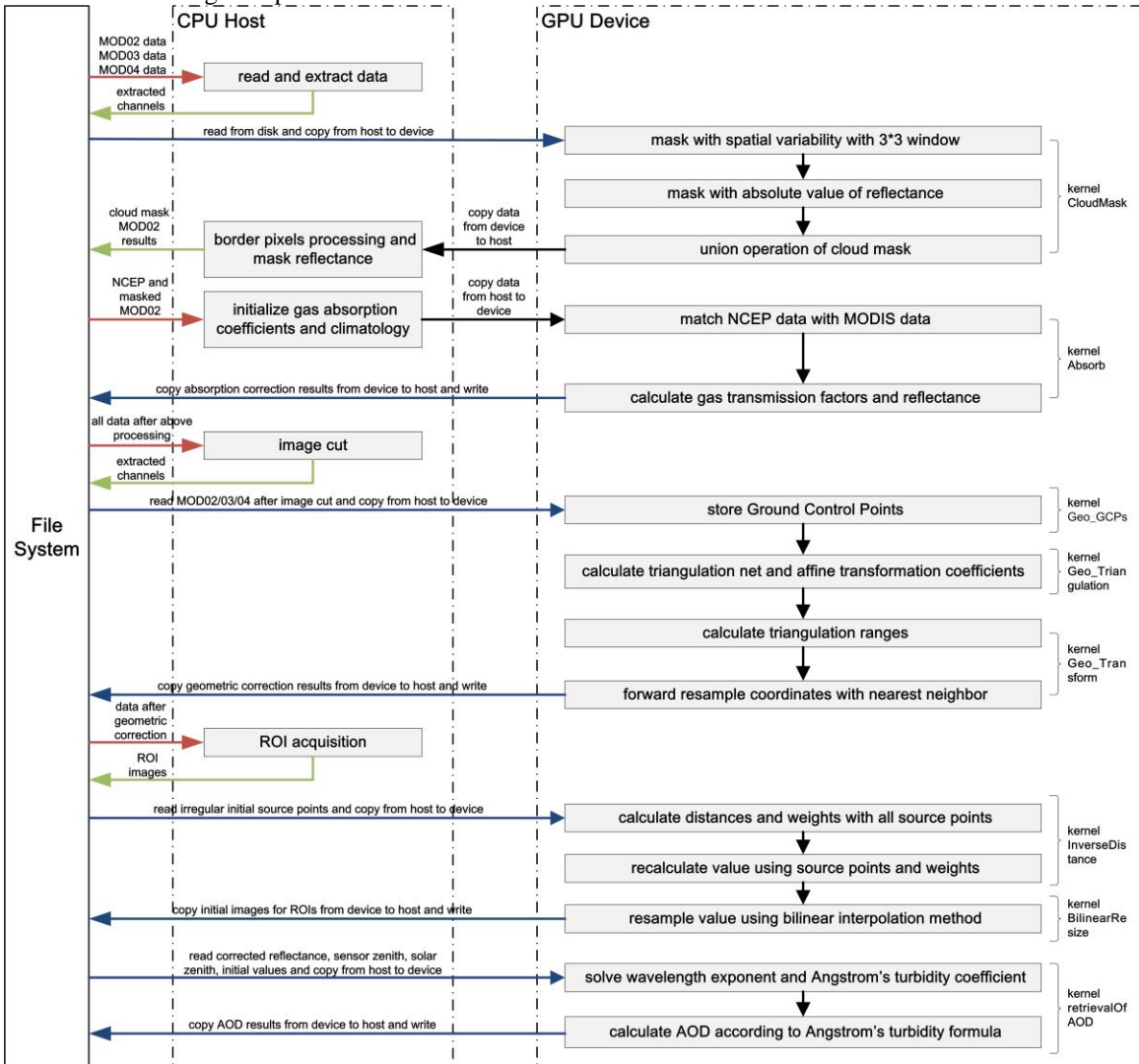


Fig. 6. Flowchart of GPU accelerated AOD retrieval procedure using SRAP-MODIS algorithm.

Algorithm 1: Pseudo code for the workflow parallelism implemented with MPI.

```

1 MPI variable declarations and initialization;
2 Call MPI_Init, MPI_Comm_size, MPI_Comm_rank, MPI_Get_processor_name;
3 if rank == size - 1 then
4   Read the job list and each job's retrieval parameter descriptions of date, latitude, longitude;
5   Read each job's, i.e. a DAG's descriptions of tasks and dependency relationships;
6   Perform the extended HEFT algorithm and store results in a in struct array schedule[job][task];
7   for index_proc = 0 to num_machines - 1 by step 1 do
8     Generate and sort a job_task_list[index_proc] for each processor from schedule[job][task];
9     if job_task_list[index_proc].size > 0 then
10      for index_job_task = 0 to job_task_list[index_proc].size - 1 by step 1 do
11        Call MPI_Send to send job_task_list[index_proc][index_job_task] to the index_proc-th processor;
12      end
13    else
14      Call MPI_Send to send finish signal to the index_proc-th processor;
15    end
16  end
17  while job_master is true do
18    Call MPI_Recv to receive finish_signal from MPI_ANY_SOURCE processor;
19    set k = status.MPI_SOURCE;
20    if all tasks on k-th processor finish then
21      Call MPI_Send to send finish signal to the k-th processor;
22    end
23    Set the numbers of processor that has finished work count_recv ++;
24    if count_recv == num_machines then
25      Set job_master as false and stops the master process;
26    end
27  end
28 else
29   while job_worker is true do
30     Call MPI_Recv to receive start_signal and num_tasks from the master;
31     if num_tasks > 0 then
32       for index_task = 0 to num_tasks - 1 by step 1 do
33         Call MPI_Recv to receive the task parameters from the master and push to the slave_task_list;
34       end
35       Connect the database;
36       for index_task = 0 to num_tasks - 1 by step 1 do
37         Query database to check the finish status of the immediate predecessor task;
38         Sleep until the the immediate predecessor tasks finish;
39         Call the workflow main function in the extern .cu file to execute the task;
40         Update the task status in database;
41       end
42       Call MPI_Send to send the finish signal to the master;
43       Set job_worker as false and stops the slave process;
44     else
45       Call MPI_Send to send the finish signal to the master;
46       Set job_worker as false and stops the slave process;
47     end
48   end
49 end
50 Call MPI_Finalize;

```

Algorithm 2: The extended HEFT scheduling approach for multiple DAGs.

```

1 Push jobs into queues according to the priorities;
2 for index_queue = 0 to queue_num - 1 by step 1 do
3   for index_dag = 0 to queue[index_queue].DAGNum - 1 by step 1 do
4     Set the computation costs of tasks and communication costs of edges with mean values;
5     Compute ranku for all tasks by traversing its graph upward, starting from the exit task;
6     for index_task = 0 to DAG[index_dag].taskNum - 1 by step 1 do
7       Select the first task ni of DAG[index_dag] from the list for scheduling;
8       for each processor pk in the processor-set (pk ∈ Q) do
9         Compute EFT(ni, pk) using the insertion-based policy among all scheduled DAGs' tasks;
10      end
11      Assign task ni to the processor pj that minimizes the EFT of task ni;
12    end
13  end
14 end

```

IV. RESULTS AND ANALYSIS

A. Experiment setup

The configurations of master and slave nodes on the GPU cluster are shown in Table I. The MPI implementations were compiled with the OpenMPI 1.8.3 and GNU gcc 4.9.1 with the ‘-O2’ flag. The GPU parallel implementations have been compiled with compute capability 3.5. The MODIS satellite data for experiments covers April 2012 and a spatial range of 35°E – 150°E, 15°N – 60°N at 1 km spatial resolution, which can be downloaded from Level 1 and Atmosphere Archive and Distribution System web.

TABLE I
EXPERIMENTAL SYSTEM CONFIGURATIONS.

	Master node	Slave node
<i>CPU configuration</i>	Intel Xeon E5-2660 server CPUs running at 2.2 GHz 8 physical cores 32GB memory memory bandwidth of 76.8 GB/s	Intel Xeon E5-1650 v2 CPU running at 3.50GHz 6 physical cores 64GB memory memory bandwidth of 51.2GB/s
<i>GPU configuration</i>	/	NVIDIA Tesla K40 2880 cores 12GB memory memory bandwidth of 288GB/s

B. Performance analysis

1) AOD retrieval results

Fig. 7 shows an example of daily AOD retrieval results for MODIS data from both TERRA and AQUA satellites using the presented framework. The retrieval accuracy has been validated in [10, 33, 34].

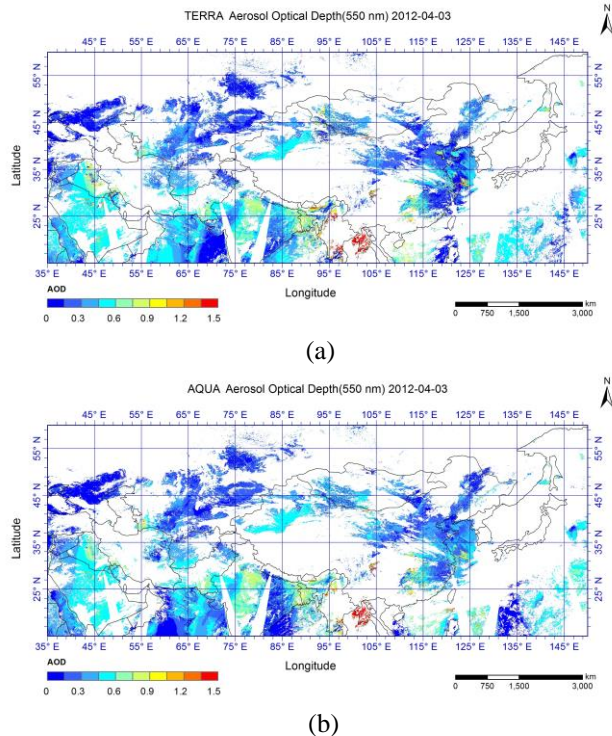


Fig. 7. An example of AOD retrieval results: (a) AOD retrieval results from TERRA MODIS data; (b) AOD retrieval results from AQUA MODIS data.

2) Fine-grained parallel performance

Fig. 8 shows the performance comparison of calculation runtime and overall runtime including data I/O for the five major processing steps when running with 1 thread on a CPU (referred to as ‘serial’ in Fig. 8) and an NVIDIA Tesla K40 GPU respectively, taking a daily AOD retrieval on 3rd April 2012 for example. The calculation and overall speedups of GPU accelerated implementations compared with the ‘serial’ version are presented in Fig. 9. The data interpolation and AOD retrieval are the most time-consuming parts with low data I/O fractions, which take 1797.50s and 967.57s for calculation, and 1798.28s and 970.66s in total when running with in serial. The GPU parallel implementation reduces the runtime to 46.63s and 8.21s for calculation, and 48.12s and 13.02s including the data I/O. The calculation and overall speedups on the target GPU range from 7.96X to 117.81X, and 1.05X to 74.53X respectively shown in Fig. 9 compared to the serial implementation counterpart. The overall runtime for the example workflow with and without GPU parallelism are 3150.27s and 499.20s, respectively. The overall speedup achieved for the workflow comes up to 6.31X.

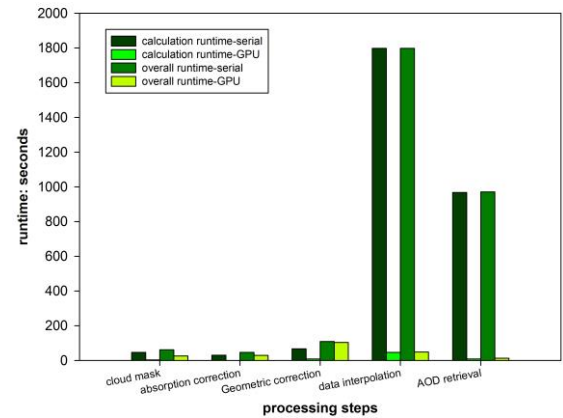


Fig. 8. Calculation and overall runtime of the serial and GPU accelerated processing.

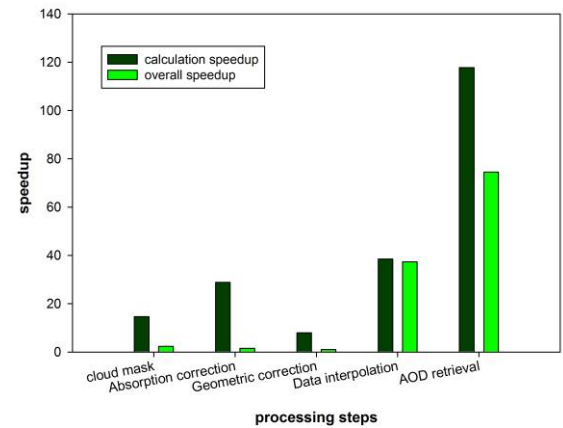


Fig. 9. Calculation and overall speedup of the GPU accelerated processing.

3) Coarse-grained parallel performance

The estimated runtime is first validated in this work. 100 runtime log samples were collected and randomly separated into 80 samples for fitting and 20 samples for validation. 14

models were obtained and used to estimate the runtime for the corresponding steps in Fig. 1. Fig. 10 presents statistical metrics including the normalized root mean square error (NRMSE), the mean absolute percent error (MAPE) and the correlation coefficient (R). The runtime estimation achieves relatively high accuracy with low MAPEs around 10%, low NRMSE ranging from 0.22 to 0.40, and R higher than 0.97 for all steps.

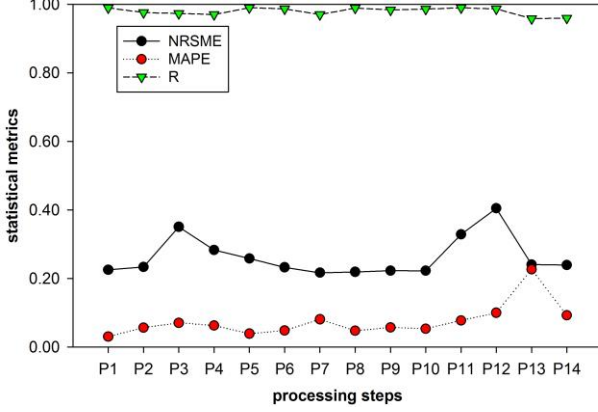


Fig. 10. Runtime estimation accuracy for steps P1-P14.

The performance of parallelism of functional tasks based on the HEFT algorithm is presented first for a daily AOD retrieval in Fig. 4. The randomly selected dataset is the same with that in the subsection fine-grained parallel performance. The retrieval workflow was scheduled to two slave nodes, i.e. p1 and p2 compared to the serial one. The overall runtime of the AOD retrieval is reduced from 3150.27s to 499.20s by the GPU parallelism and further to 309.75s by the functional parallelism on the GPU cluster shown in Fig. 4. Speedups of 1.61X compared with the GPU parallel version and 10.17X compared with the original serial retrieval workflow are obtained. However, an idle slot of 98.92s is also observed due to the data dependency between the task P9 and P8.

The proposed scheduling strategy is performed to retrieve the AOD time series product of one month as described in the subsection of experiment setup, and the coarse-grained parallelism is evaluated in terms of the average makespan, overall makespan and the impacts of priorities. The metrics considered are defined as follows:

- *Average makespan*: the makespan of a workflow is the period between its arrival and finish, and the average makespan $makespan_{average}$ for the first N workflows after scheduling all the M workflows ($1 \leq N \leq M$, $M=30$ in this paper), is defined as [44]:

$$makespan_{average} = \frac{1}{N} \sum_{k=0}^{N-1} makespan_{p_k} \quad (6)$$

- *Overall makespan*: The overall makespan for workflows already scheduled is defined as [44]:

$$makespan_{overall} = \max_{p_k \in DAGs} makespan_{p_k} \quad (7)$$

Considering all retrieval jobs with the same priorities, the resulting average makespan and overall makespan according

to the number of workflows is shown in Fig. 11 and Fig. 12. The makespan evaluates the scheduling performance in the light of duration of workflow execution. The parallel method reduces the average makespans from 10.69s to 68.53s, which leads to a performance improvement of 2.26% to 28.65% for the first 9 workflow, and the overall makespans reduce from 11.85s to 68.53s, with a performance improvement of 1.39% to 28.65% for the first 11 workflows. The performance gains decrease as the workflow number further increasing on the benchmarked GPU cluster with limited computing nodes. It is reasonable that there are dependencies among tasks of a workflow; idle slots exist when workflows are parallelized on a GPU cluster, which however might be never used since they are too small to insert any later tasks. This performance degradation effect will be eliminated if there are sufficient computing resources. Table II shows the comparison of the average makespan when randomly setting the retrieval jobs with different priorities. It can be observed that the average makespan of the jobs with higher priorities can benefit from the presented coarse-grained parallel method. For instance, the average makespan of retrieval jobs reduces from 220.95s to 157.04s at level-1 and from 311.55s to 262.47s at level-2.

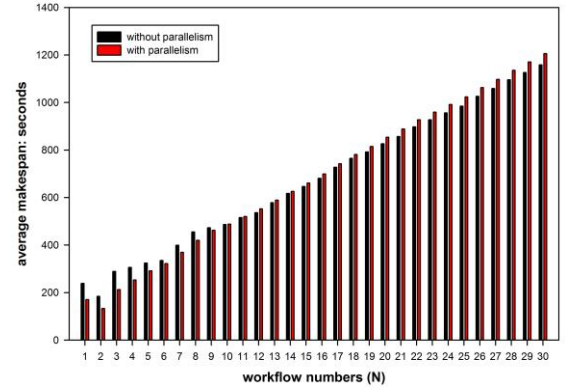


Fig. 11. Average makespan along with the number of workflows.

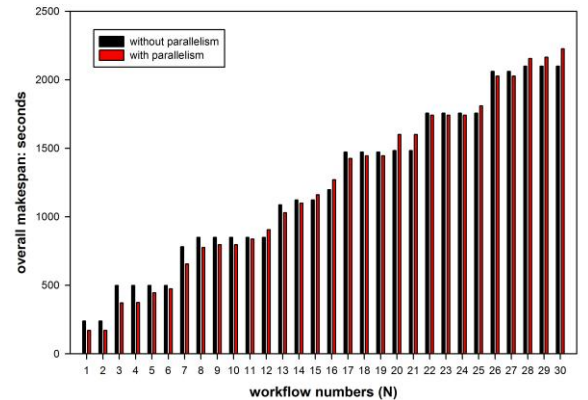


Fig. 12. Overall makespan along with the number of workflows.

TABLE II
THE IMPACTS OF PRIORITIES ON AVERAGE MAKESPAN.

	PRIORITY		
	Level-1	Level-2	Level-3
<i>Presented method</i>	157.04s	262.47s	839.70s
<i>Serial workflow</i>	220.95s	311.55s	269.59s

V. CONCLUSION

In this paper we design and implement a high performance framework to accelerate the time-consuming RS time-series quantitative retrieval issue on a GPU cluster, taking the AOD retrieval from satellite data as a study case. The presented framework exploits the multi-level parallelism for enhancing efficiency. For the coarse-grained parallelism, the AOD time series retrieval is represented as multi-DAG workflows and scheduled based on a list-based heuristic algorithm HEFT with estimated runtime, taking the idle slot and priorities into consideration. For the fine-grained parallelism, the parallel strategies for major remote sensing image data processing algorithms divided into three categories, i.e. the point or pixel-based operations, the local operations and the global or irregular operations are summarized. The parallel framework has been implemented based on MPI and CUDA.

Experimental results with the AOD retrieval study case show that the GPU parallelism has achieved a high speedup of 117.81X for calculation and 74.53X including data I/O for the time-consuming AOD retrieval step, and as well a high overall speedup of 6.31X for the overall retrieval workflow. With the estimated runtime with relatively high accuracy, the coarse-grained functional parallelism of a retrieval job further achieves a speedup of 1.61X compared with the GPU parallel version. Furthermore, a performance improvement up to a maximum of 28.65% is observed for the makespan with sufficient resources for retrieving the AOD time series which can be represented as multi-DAGs. The experimental results with the AOD retrieval case verify the effectiveness of the presented framework.

Despite the performance improvement obtained for the AOD retrieval case, our work can also provide the parallel computing suggestions for other time series quantitative RS retrieval applications with similar patterns on a GPU cluster platform, since these category applications follow the same coarse and fine grained multi-level parallelism. As for the coarse-grained parallelism, the runtime estimation and task scheduling approaches can be directly applied with different inputs. Meanwhile for the fined-grained parallelism, we have summarized the common RS image data processing patterns, i.e. the pixel, local, global or irregular operations, and also given the parallel implementations on GPUs.

At present, the computing nodes equipped with GPU s are taken as basic processing units for the task scheduling at the coarse-grained parallelism, and a hybrid implementation of

MPI-CUDA is realized to promote efficiency. However, the CPU cores of multicore processors on each node are not fully utilized, thus in future work the presented framework will be further optimized in terms of the performance estimation and co-scheduling of CPU and GPU, as well as its hybrid parallel implementations.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions on this paper. MODIS data were available through NASA MODIS LAADS.

REFERENCES

- [1] J. Fan, J. Yan, Y. Ma, and L. Wang, "Big Data Integration in Remote Sensing across a Distributed Metadata-Based Spatial Infrastructure," *Remote Sensing*, vol. 10, p. 7, 2018.
- [2] E. Masuoka, C. Tilmes, N. Devine, G. Ye, and M. Tilmes, "Evolution of the MODIS science data processing system," in *IEEE 2001 International Geoscience and Remote Sensing Symposium*, 2001, pp. 1454-1457.
- [3] L. Liu, Y. Xue, G. Jie, and J. Liu, "Description of an ontology-based remote sensing model service with an integrated framework environment for remote sensing applications," *Remote Sensing Letters*, vol. 6, pp. 804-813, 2015.
- [4] S. Liang, *Quantitative Remote Sensing of Land Surfaces*: Wiley-Interscience, 2004.
- [5] J. Verbesselt, R. Hyndman, G. Newnham, and D. Culvenor, "Detecting trend and seasonal changes in satellite image time series," *Remote Sensing of Environment*, vol. 114, pp. 106-115, 2010.
- [6] H. Luo, C. Liu, C. Wu, and X. Guo, "Urban change detection based on Dempster-Shafer theory for multitemporal very high-resolution imagery," *Remote Sensing*, vol. 10, p. 980, 2018.
- [7] Y. Luo, X. Zheng, T. Zhao, and J. Chen, "A climatology of aerosol optical depth over China from recent 10 years of MODIS remote sensing data," *International Journal of Climatology*, vol. 34, pp. 863-870, 2014.
- [8] L. A. Remer, R. G. Kleidman, R. C. Levy, Y. J. Kaufman, D. Tanré, S. Mattoo, J. V. Martins, C. Ichoku, I. Koren, and H. Yu, "Global aerosol climatology from the MODIS satellite sensors," *Journal of Geophysical Research: Atmospheres*, vol. 113, 2008.
- [9] C. Yang, Y. Xu, and D. Nebert, "Redefining the possibility of digital Earth and geosciences with spatial cloud computing," *International Journal of Digital Earth*, vol. 6, pp. 297-312, 2013.
- [10] J. Liu, D. Feld, Y. Xue, J. Garcke, T. Soddemann, and P. Pan, "An efficient geosciences workflow on multi-core processors and GPUs: a case study for aerosol optical depth retrieval from MODIS satellite data," *International Journal of Digital Earth*, vol. 9, pp. 748-765, 2016.
- [11] P. Baumann, P. Mazzetti, J. Ungar, R. Barbera, D. Barboni, A. Beccati, L. Bigagli, E. Boldrini, R. Bruno, and A. Calanducci, "Big Data Analytics for Earth Sciences: the EarthServer approach," *International Journal of Digital Earth*, vol. 9, pp. 3-29, 2016.
- [12] C. A. Lee, S. D. Gasster, A. Plaza, C. I. Chang, and B. Huang, "Recent Developments in High Performance Computing for Remote Sensing: A Review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, pp. 508-527, Sep 2011.
- [13] C. Gonzalez, S. Sanchez, A. Paz, J. Resano, D. Mozos, and A. Plaza, "Use of FPGA or GPU-based architectures for remotely sensed hyperspectral image processing," *Integration-the Vlsi Journal*, vol. 46, pp. 89-103, Mar 2013.
- [14] A. Paz and A. Plaza, "Clusters versus GPUs for parallel target and anomaly detection in hyperspectral images," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, pp. 1-18, 2010.
- [15] Y. Xue, J. W. Ai, W. Wan, H. D. Guo, Y. J. Li, Y. Wang, J. Guang, L. L. Mei, and H. Xu, "Grid-enabled high-performance quantitative aerosol retrieval from remotely sensed data," *Computers & Geosciences*, vol. 37, pp. 202-206, Feb 2011.
- [16] W. Guo, J. Y. Gong, W. S. Jiang, Y. Liu, and B. She, "OpenRS-Cloud: A remote sensing image processing platform based on cloud computing

- environment," *Science China Technological Sciences*, vol. 53, pp. 221-230, 2010.
- [17] S. Bernabé, S. Sánchez, A. Plaza, S. López, J. A. Benediktsson, and R. Sarmiento, "Hyperspectral Unmixing on GPUs and Multi-Core Processors: A Comparison," *IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing*, vol. 6, pp. 1386-1398, 2013.
- [18] P. Quesada-Barriuso, F. Arguello, D. B. Heras, and J. A. Benediktsson, "Wavelet-Based Classification of Hyperspectral Images Using Extended Morphological Profiles on Graphics Processing Units," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, pp. 2962-2970, Jun 2015.
- [19] Y. L. Chang, J. N. Liu, Y. L. Chen, W. Y. Chang, T. J. Hsieh, and B. M. Huang, "Hyperspectral band selection based on parallel particle swarm optimization and impurity function band prioritization schemes," *Journal of Applied Remote Sensing*, vol. 8, Aug 2014.
- [20] S. Bernabé, S. Lopez, A. Plaza, and R. Sarmiento, "GPU Implementation of an Automatic Target Detection and Classification Algorithm for Hyperspectral Image Analysis," *IEEE Geoscience & Remote Sensing Letters*, vol. 10, pp. 221-225, 2013.
- [21] U. Bhangale, S. S. Durbha, R. L. King, N. H. Younan, and R. Vatsavai, "High performance GPU computing based approaches for oil spill detection from multi-temporal remote sensing data," *Remote Sensing of Environment*, 2017.
- [22] R. Giordano and P. Guccione, "ROI-Based On-Board Compression for Hyperspectral Remote Sensing Images on GPU," *Sensors*, vol. 17, 2017.
- [23] L. Y. Fang, M. Wang, D. R. Li, and J. Pan, "CPU/GPU near real-time preprocessing for ZY-3 satellite images: Relative radiometric correction, MTF compensation, and geocorrection," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 229-240, Jan 2014.
- [24] M. Abouali, J. Timmermans, J. E. Castillo, and B. Z. Su, "A high performance GPU implementation of Surface Energy Balance System (SEBS) based on CUDA-C," *Environmental Modelling & Software*, vol. 41, pp. 134-138, Mar 2013.
- [25] X. Su, J. Wu, B. Huang, and Z. Wu, "GPU-Accelerated Computation for Electromagnetic Scattering of a Double-Layer Vegetation Model," *IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing*, vol. 6, pp. 1799-1806, 2013.
- [26] J. Mielikainen, B. Huang, and H. L. A. Huang, "GPU-Accelerated Multi-Profile Radiative Transfer Model for the Infrared Atmospheric Sounding Interferometer," *IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing*, vol. 4, pp. 691-700, 2011.
- [27] E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer. (2018). *TOP 500 LIST*. Available: <https://www.top500.org/lists/2018/06/>
- [28] A. Agathos, J. Li, D. Petcu, and A. Plaza, "Multi-GPU Implementation of the Minimum Volume Simplex Analysis Algorithm for Hyperspectral Unmixing," *IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing*, vol. 7, pp. 2281-2296, 2014.
- [29] M. A. Hossam, H. M. Ebied, M. H. Abdel-Aziz, and M. F. Tolba, "Accelerated hyperspectral image recursive hierarchical segmentation using GPUs, multicore CPUs, and hybrid CPU/GPU cluster," *Journal of Real-Time Image Processing*, pp. 1-20, 2014.
- [30] Z. Lei, M. Wang, D. R. Li, and T. L. Lei, "Stream Model-Based Orthorectification in a GPU Cluster Environment," *IEEE GEOSCIENCE AND REMOTE SENSING LETTERS*, vol. 11, pp. 2115-2119, Dec 2014.
- [31] Y. Ma, H. P. Wu, L. Z. Wang, B. M. Huang, R. Ranjan, A. Zomaya, and W. Jie, "Remote sensing big data computing: Challenges and opportunities," *Future Generation Computer Systems-the International Journal of Grid Computing and Esience*, vol. 51, pp. 47-60, Oct 2015.
- [32] F. Huang, Y. Chen, L. Li, J. Zhou, J. Tao, X. Tan, and G. Fan, "Implementation of the parallel mean shift-based image segmentation algorithm on a GPU cluster," *International Journal of Digital Earth*, vol. 12, pp. 328-353, 2019.
- [33] Y. Xue, X. He, H. Xu, G. Jie, J. Guo, and L. Mei, "China Collection 2.0: The aerosol optical depth dataset from the synergetic retrieval of aerosol properties algorithm," *Atmospheric Environment*, vol. 95, pp. 45-58, 2014.
- [34] J. Liu, D. Feld, Y. Xue, J. Garcke, and T. Soddemann, "Multicore Processors and Graphics Processing Unit Accelerators for Parallel Retrieval of Aerosol Optical Depth From Satellite Data: Implementation, Performance, and Energy Efficiency," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, pp. 2306-2317, May 2015.
- [35] Y. Ma, L. J. Chen, P. Liu, and K. Lu, "Parallel programing templates for remote sensing image processing on GPU architectures: design and implementation," *Computing*, vol. 98, pp. 7-33, Jan 2016.
- [36] Y. J. Kaufman, D. Tanré, and O. Boucher, "A satellite view of aerosols in the climate system," *Nature*, vol. 419, pp. 215-23, 2002.
- [37] T. Stocker, D. Qin, G. Plattner, M. Tignor, S. Allen, J. Boschung, A. Nauels, Y. Xia, V. Bex, and P. Midgley, "IPCC, 2013: Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change," in *of the Intergovernmental Panel on Climate Change*, 2013, pp. 710-719.
- [38] Y. Xie, Y. Xue, Y. Che, G. Jie, L. Mei, D. Voorhis, C. Fan, L. She, and H. Xu, "Ensemble of ESA/AATSR Aerosol Optical Depth Products Based on the Likelihood Estimate Method With Uncertainties," *IEEE Transactions on Geoscience & Remote Sensing*, vol. PP, pp. 1-11, 2017.
- [39] Y. XUE and A. P. CRACKNELL, "Operational bi-angle approach to retrieve the Earth surface albedo from AVHRR data in the visible band," *International Journal of Remote Sensing*, vol. 16, pp. 417-429, 1995.
- [40] C. B. Lee, Y. Schwartzman, J. Hardy, and A. Snavely, "Are User Runtime Estimates Inherently Inaccurate?," in *International Conference on Job Scheduling Strategies for Parallel Processing*, 2004, pp. 253-263.
- [41] A. W. Mu'Alem and D. G. Feitelson, "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling," *Parallel & Distributed Systems IEEE Transactions on*, vol. 12, pp. 529-543, 2001.
- [42] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 260-274, 2002.
- [43] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Task scheduling algorithms for heterogeneous processors," in *Heterogeneous Computing Workshop*, 2002, pp. 3-14.
- [44] L. F. Bittencourt and E. R. M. Madeira, "Towards the Scheduling of Multiple Workflows on Computational Grids," *Journal of Grid Computing*, vol. 8, pp. 419-441, 2010.
- [45] A. Wijayasiri, T. Banerjee, S. Ranka, S. Sahni, and M. Schmalz, "Dynamic Data-Driven SAR Image Reconstruction Using Multiple GPUs," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, pp. 4326-4338, 2018.
- [46] V. Volkov, "Better performance at lower occupancy," in *Proceedings of the GPU technology conference, GTC*, 2010, p. 16.
- [47] Z. Wu, Q. Wang, A. Plaza, J. Li, J. Liu, and Z. Wei, "Parallel implementation of sparse representation classifiers for hyperspectral imagery on GPUs," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, pp. 2912-2925, 2015.